Implementation of an Extended Prediction Self-Adaptive Controller using LabVIEW^{TM}

Silviu Folea, George Mois, Cristina I. Muresan, Liviu Miclea
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Silviu.Folea, George.Mois, Cristina.Pop, Liviu.Miclea@aut.utcluj.ro

Robain De Keyser
Department of Electrical Energy,
Ghent University,
Ghent, Belgium
Robain.DeKeyser@UGent.be

Marcian Cirstea
Department of Computing and Technology,
Anglia Ruskin University,
Cambridge, UK
marcian.cirstea@anglia.ac.uk

***Abstract* - The implementation of the Extended Prediction Self-Adaptive Controller is presented in this paper. It employs LabVIEW^{TM} graphical programming of industrial equipment and it is suitable for controlling fast processes. Three different systems are used for implementing the control algorithm. The research regarding the controller design using graphical programming demonstrates that a single advanced control application can run on Windows, real time operating systems and FPGA targets without requiring significant program modifications. The most appropriate device may be selected according to the required processing time of the control signal and of the application. A relevant case study is used to exemplify the procedure.**

Keywords—Field programmable gate arrays, Predictive control, Benchmark testing, Real-time systems.

I. INTRODUCTION

Predictive control has emerged as one of the most commonly researched control algorithms in various fields of industrial activity. Predictive control, in comparison to other modern control methods, has a series of features that make it appealing to both the researcher, as well as the industrial engineer, such as: intuitive principles, performance oriented design parameters, intrinsic ability for handling time delays and nonlinearities, as well as different constraints (ranging from actuator/sensor constraints to safety or quality constraints). Typically, predictive control has been used in the control of processes with large time constants and time delays, such as chemical plants [1]. However, an increasing number of model predictive control applications have been reported in the control of fast dynamical systems [2], [3]. Traditionally, the main limitation of MPC (Model Predictive Control) resulted from the long computational time needed for determining the optimal control action. This represents one of the major reasons for which MPC controllers are used extensively only in industrial computers that could manage the complexity of the optimization. Nonetheless, the use of DSPs and FPGAs in control application has nowadays led to the reduction of the time needed for solving the constrained optimization problem with a period of tens or hundreds of microseconds [4].

The purpose of this paper is to present an efficient and robust control solution, based on a specific version of the MPC control algorithm, for fast dynamic systems, using FPGA devices and the LabVIEW^{TM} graphical programming environment. The choice for graphical programming in terms of implementing the proposed control solution is based on a more user-friendly configuration environment and a very short project development time, compared to hardware description languages such as VHDL [5]. A simple application for a DC motor was chosen for validation and testing purposes. The DC motor is part of the vacuum pumps used to maintain an efficient thermal isolation in the vacuum jacket of a train of three carbon isotopes separation columns. These need to be carefully controlled since a failure of the vacuum leads to the compromise of the entire separation process [6], [7]. Additionally, the DC motor supports a wide range of command rates and execution time variations, without being damaged or broken. The paper presents three different implementations of the proposed control solution, a PC based control system, one running on a real-time target and one based on an FPGA. Comparisons between these three different implementations show the efficiency of the proposed solution.

The paper is structured into seven parts. The advantages offered by FPGA technology in control applications are highlighted in the second section, while the third one describes the major concepts related to the proposed MPC method, the EPSAC (Extended Prediction Self Adaptive Controller) control algorithm. Section IV describes the design of the EPSAC controller, as well as the methodology used for the FPGA implementation. Information regarding the details of the hardware and of the software setup is detailed in section five, with the testing, validation and performance evaluation of the proposed solution being synthesized in the sixth section. The final section contains the concluding remarks.

II. FPGA-BASED PROCESS CONTROL APPLICATIONS

Industrial control systems applications can benefit from the advantages brought by FPGA technology as compared to traditional microcontroller and Digital Signal Processor-based solutions (DSP). Some of these advantages consist in their ability to provide increased levels of performance in terms of throughputs and bandwidth, while maintaining reduced cost and dimensions of the developed equipment, and making the achievement of high energy efficiency and reliability possible [8], [9], [10]. Since the control algorithms are getting more and more complex, the inherent parallelism within FPGAs and their ever-increasing resource density makes them very attractive for industrial applications [11]. Being fully programmable, the logic blocks and the interconnections that make up an FPGA chip can implement solutions entirely tailored for specific control algorithms. Often including powerful hard processors, or allowing the implementation of IP soft cores for efficient 32-bit RISC processors, these customizable Systems-on-chip (SoCs) can implement control loops running at frequencies higher than one MHz [12]. The inclusion of logic, such as CPUs, RAM, and buses, within a single chip and, consequently, the simplification of the printed circuit board (PCB), lead to cheaper solutions even if FPGAs are more expensive than their main counterparts, microcontrollers and DSPs. A systematic comparison between the two predominant devices used in digital controllers, namely FPGAs and DSPs, can be found in [13]. Another important feature of Field Programmable Gate Arrays is the possibility of in-the-field configuration, which eases the controller modification process in case this is desired. Furthermore, they can be dynamically reconfigured, providing the controller with the possibility of adapting to the needs of the plant and to the changes in the environment [8].

Some of the applications in the field of electrical systems built around reconfigurable chips include reliable low-complexity reusable digital controllers [14], adaptive digital PI controllers [15], communication processors and interfaces, signal processors [16] and many others. Model predictive control was applied to power converters [17] and induction machines [18], papers [19] and [20] showing that solutions implemented in FPGAs offer good control performances.

III. EPSAC Control Principles

The EPSAC methodology is based on the typical approach of Model Based Predictive Controllers that use an on-line process model to compute the predicted process output with the purpose of optimizing the future control actions. The optimal control action generated by the EPSAC controller is based upon the minimization of a cost function, represented as the error signal between the specified reference trajectory and a future predicted process output, as well as the control effort required to eliminate the error [21]:

$$\sum_{k=N_1}^{N_2}\left[r(t+k/t)-y(t+k/t)\right]^2+\lambda\sum_{k=0}^{N_u-1}\left[\Delta u(t+k/t)\right]^2 \quad (1)$$

where N_2 and N_1 are the maximum and the minimum prediction horizons, N_u is the control horizon, λ is a weight parameter. The choice for the maximum and minimum prediction horizons usually plays an important role in the minimization of the cost index (1). The signals involved in the cost function given in (1) are the measured process output, denoted *y(t)*, the process input *u(t)* and the reference trajectory *r(t)*. The control signal in (1) is given by [21]:

$$\Delta u(t+k/t)=u(t+k/t)-u(t+k-1/t)\text{, with (2)}$$

$$\Delta u(t+k/t)\equiv 0\text{, for } k\geq N_u\text{. (3)}$$

The EPSAC control methodology uses previous measurements of the process output and input signals, as well as some future values of the input signal to predict the process output. Thus, the generic model given in (4) can be used for predicting the process output:

$$y(t+k/t)=x(t+k/t)+n(t+k/t)\text{, (4)}$$

where *x(t)* represents the process model output, while *n(t)* is the process disturbance. In computing the process output as indicated in (4), the process model output $x(t + k|t)$ needs to be predicted according to an existing model of the process. For the prediction of the disturbance signal $n(t + k|t)$ filtering techniques are usually employed.

Assuming the process model for a single-input-single-output system is given by:

$$x(t)=\frac{B(q^{-1})}{A(q^{-1})}u(t)\text{, (5)}$$

the output model *x* may be predicted *k* samples ahead using previous values of the process model and of the control input *u*, considering that polynomials $B(q^{-1})$ and $A(q^{-1})$ in (5) are fully known.

The manipulated variables are computed then as the sum of a basic future control scenario, called $u_{base}(t + k|t), k \geq 0$ and of an optimizing future control action $\delta_u(t +k|t),\ 0 \leq k \leq N_u - 1$. The optimizing future control actions are computed as the optimal solution of [24]:

$$\mathbf{U}^*=(\mathbf{G}^\mathbf{T}\mathbf{G})^{-\mathbf{1}}\mathbf{G}^\mathbf{T}(\mathbf{R}-\mathbf{Y})\text{. (6)}$$

As a result of the optimization, only the first element in $\mathbf{U}^*$, denoted $\delta_u(t|t)$, is used to update the control signal as indicated below:

$$u(t)=u_{base}(t/t)+\delta_u(t/t)\text{. (7)}$$

At the next sampling instant, the new measured output signal value $y(t + 1)$ is used to update again the control signal $u(t + 1)$.

IV. EPSAC DC Motor Controller

The EPSAC predictive algorithm has been successfully used for controlling a wide range of electrical systems [17]. The application example considered in this paper is a DC motor, as an integrated part of complex vacuum pumps used for maintaining an efficient thermal isolation in the vacuum jacket of a train of three carbon isotopes separation columns [6], [7]. The careful control of these DC motors indirectly ensures a safe operation of the isotope separation columns, since a failure of the vacuum pumps ultimately leads to the compromise of the entire isotope separation columns. Apart from this aspect, the DC motor provides a highly flexible stand for testing, as well as for achieving rapid performance comparisons.

The block diagram of the system is presented in Fig. 1, including the main components, namely the controller, the driver, the signal processing module, the DC motor and the load, consisting of a generator and a controlled resistive load.

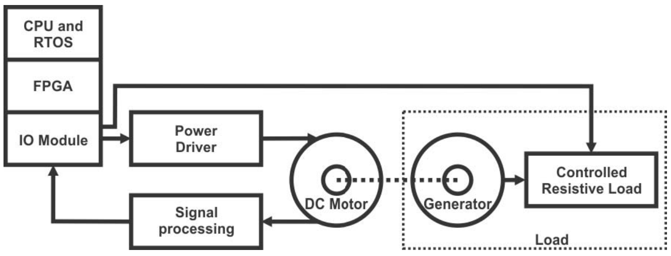

Fig. 1. System block diagram

An embedded CompactRIO™ system, easily programmable using the LabVIEW™, has been used for implementing the EPSAC control algorithm. It includes two chips: the real time controller running at 400 MHz and a chassis with a Virtex-II FPGA, as well as an input-output module, as indicated also in Fig. 1.

4.1. Modeling of the DC motor for tuning the EPSAC controller

The EPSAC control strategy implemented in the FPGA has been designed for the speed reference tracking of the DC motor. A mathematical model, described by the two polynomials $A(q^{-1})$ and $B(q^{-1})$ in (5), is firstly needed in order to properly tune the EPSAC controller. To determine the polynomials $A(q^{-1})$ and $B(q^{-1})$, experimental identification techniques were employed.

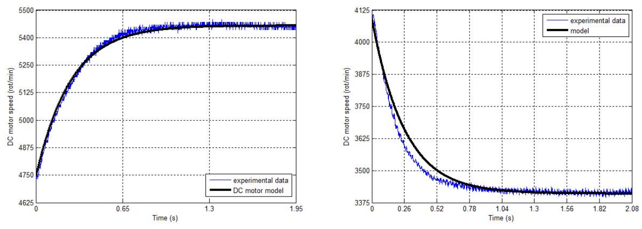

Fig. 2. Experimental data for process identification – speed rises and decreases

Such experimental data used for identification purposes are presented in Fig. 2, along with the identified process model, considering both an increase as well as a decrease in the speed of the DC motor. The output that needs to be controlled is the DC motor's rotation speed, while the control signal is the DC voltage supplied to the rotor. An initial input voltage of 70% has been first supplied to the DC motor, for the experimental identification tests in Fig. 2. Subsequently, a step input of +10% was applied to the rotor, whereas for the speed decreasing test, a -10% step change in the supplied voltage has been considered.

Based on the shape of the step response, a first order transfer function was selected to model the process, with the associated gain and time constant determined through graphical identification techniques. The polynomials $A(q^{-1})$ and $B(q^{-1})$ are computed based on the determined transfer function using the zero-order hold discretization method, with a sampling time $Ts = 0.015$ sec:

$$A(q^{-1}) = 1 - 0.94q^{-1} \ , \ B(q^{-1}) = 1.54q^{-1} . \quad (8)$$

In the EPSAC methodology, for delay free processes such as the DC motor in the case study, the minimum prediction horizon is $N_1 = 1$ sample. The maximum prediction horizon is chosen in order for the predicted signal to capture around 60% of the process dynamics [21]. Thus, for the case study considered in this paper $N_2 = 10$ samples. The cost function in (1) is further simplified to minimize solely the error signal, by taking $\lambda = 0$ and $N_u = 1$.

The controller designed with the parameters as chosen above, has been firstly tested in the MATLAB® simulation environment. Then, the EPSAC controller has been implemented in the FPGA module using the guidelines given in Sections IV and V.

4.2. FPGA implementation of EPSAC

An optimal implementation method of various control algorithms on FPGA targets, realized according to specific analysis and simulation environments, should be carried out bearing in mind the steps that follow below:

1) The code used to simulate the control algorithm in the LabVIEW™ environment on the PC or on the real-time target should be rewritten;
2) Control vectors generated during simulations should be used in the testing of the program;
3) Floating-point data should be converted to fixed-point format (FXP) or integer format (INT);
4) Implementations using the control vectors and the data available in the second step should be comparatively tested and analyzed;
5) Steps 3 and 4 above should be performed iteratively until the steady state errors are acceptable; in the case of this paper it is assumed that a small error is acceptable.

Since MATLAB® sequences of code using MathScript are not supported on the FPGA target, this procedure has been avoided.

V. Hardware and Software Implementation

Real-life control system implementation poses problems concerning both the hardware and software setups. First, the development of the hardware must take into account parameters such as the compatibility between the components used, signal conditioning, and placing and routing. Second, the most important aspect that the software must take into account is the architecture of the equipment chosen for running the application. However, the use of graphical programming, namely LabVIEW™ code, in the case of the example application presented in this paper, eases and shortens the software development process [22].

5.1. Hardware

The hardware component of the system includes two PCBs for interfacing the DC motor. The first one processes the signal acquired from the speed transducer. This action implies the amplification of the signal from the encoder, represented by the speed transducer, and the filtering and formatting of the signal for obtaining rectangular pulses. The input and output signals are presented in Fig. 3.

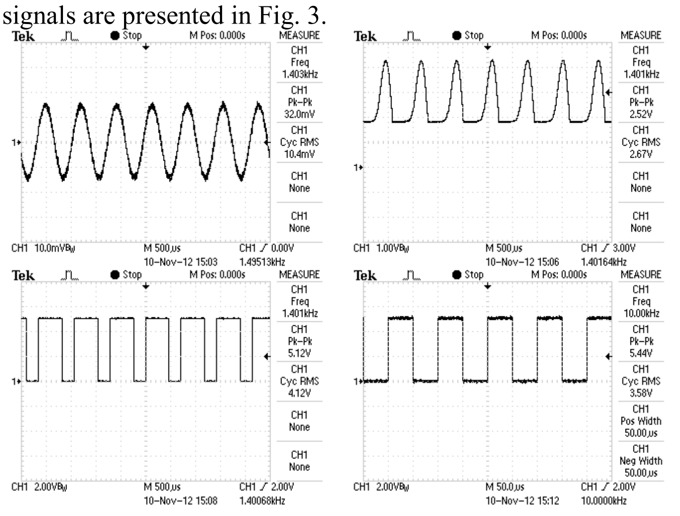

Fig. 3. Speed transducer signal (original, filtered, upper-left, and amplified, upper-right, trigger Schmitt output, lower-left, PWM command, lower-right)

This first board also includes the power driver for the motor command. A second circuit board was developed for the load of the DC motor. This is a digitally controlled resistive load represented by a motor acting as a generator. It has characteristics which are similar to the ones of the first motor and it is connected to a resistive load. Finally, a stand for verification, which includes the Programmable Automation Controller (PAC), a power supply, the DC motor and the PCBs, was built. This can be seen in Fig. 4.

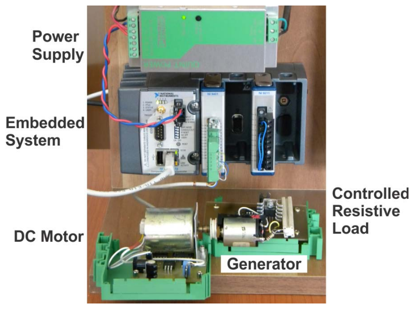

Fig. 4. The experimental stand

5.2. *Software*

The EPSAC algorithm was implemented in FPGA through the use of three *while* loops, each one of them performing a different action:

1. Measurement of motor speed;
2. Generation of the PWM for changing the speed of the motor;
3. Main loop, running the control algorithm.

The application also has a component running on the real-time target of the automation controller, which opens a connection to the FPGA program. Here, method nodes are used for setting the values of the parameters and a continuous loop is used for reading the measured values in order to display them in real time. The entire control algorithm is implemented in the FPGA and only a small amount of data, consisting in parameters and measured values, is transferred between the two components of the PAC. Fig. 5 presents the main loop of the application, the one that implements the EPSAC algorithm, and the front panel of the system.

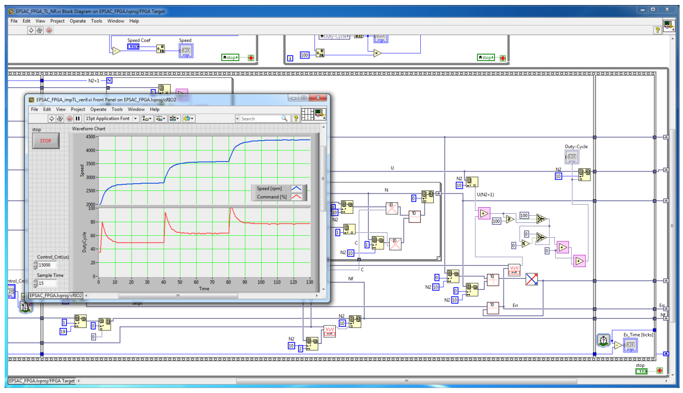

Fig. 5. LabVIEW^{TM} application

VI. Experimental Testing and Validation

In general, the FPGA design testing and validation are performed using simulator-specific environments. However, the research presented in this paper implies the development of several benchmark programs that run on various target devices in order to achieve a comparative evaluation of the computation performance achieved in each case.

The functions provided by LabVIEW^{TM} are able to access the real-time timers of the tested systems, with resolutions of milliseconds (PC), microseconds (real-time target) or tens of nanoseconds (FPGA – 25 ns), depending on the target, for the measurement of execution time and of the jitter. This later characteristic consists in the variation of the loop execution time. The benchmarks imply the development of frameworks encasing the application in order to determine the execution times and the jitter in each case. Finally, histograms for the data obtained during testing were realized.

In the first phase, the target devices selected for running the controller were: a personal computer, a real-time controller and an FPGA. The computational performance obtained after running this test are presented in Table I. It shows the maximum reachable values on each platform, pointing out that the FPGA-based EPSAC controller can be used in the case of fast dynamic processes.

TABLE I. Computational Performance

	Execution Time	**Jitter**
PC, i5 M480, Windows 7	28 µsec	0.2 ... 3.2 msec
Real-Time Target PowerPC with RTOS	2.8 msec	1.8 msec
FPGA Virtex-II	5 µsec	≤ 25 nsec

For being able to run the first benchmark program on a general purpose computer, the EPSAC algorithm code was compiled on a PC and run locally. The testing of the performances in the second case required the transfer of the benchmark to a real-time controller. The third set of tests was performed on an FPGA, employing different implementation options provided by the graphical programming environment. One of the most important problems encountered in the implementation of the algorithm consists in the data representation, each target allowing only specific settings for this parameter. The personal computer and the real-time target allow floating-point, double type data representation (DBL), while the FPGA supports only fixed-point. This is why a compromise between data accuracy and FPGA resource usage has to be made. Thus, the difficulty here lies in the choosing of the proper format for fixed-point data, consisting in the integer word length and in the entire word length, and in the computations having arrays as operands. For achieving high execution speeds on the FPGA target, specially designed mathematical operations, which allow the specification and configuration of data representations, were used for both inputs and outputs. Table I shows that the algorithm implemented in the FPGA target runs more than 5 times faster than the one deployed on the PC, achieving the same control performance.

Being a hardware implementation, the amount of occupied resources varies depending on a wide range of factors, such as data representation and applied optimizations, directly influencing the power consumption. It should be noted that the area occupied by the controller is specific to the LabVIEW^{TM} implementation and can differ from one using HDLs. The device and the software version used for generating the configuration also affect the used resource count. However, if the dynamics of the process permits it, the clock frequency can be decreased for reducing the energy consumption. The example application presented in this paper benefits from this, the extension of the execution time leaving the control unchanged. Another major advantage of the proposed approach is represented by the short application development time, offered by the use of graphical programming.

A vector containing a sequence of control input values, as well as a second vector containing the corresponding output signal values were used for evaluating the accuracy of the proposed solution. Fig. 6 shows the closed loop experimental results in comparison to the simulation results.

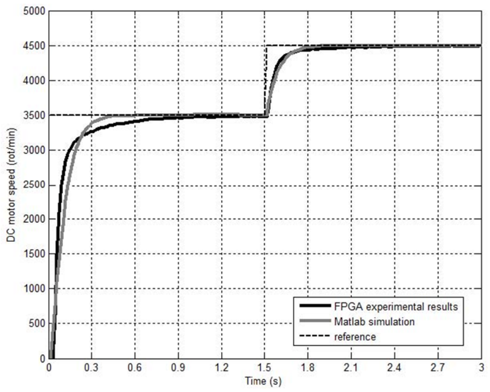

Fig. 6. Comparison between simulation and experimental data - Output amplitude (rot/min)

Both the experimental and simulated results show a closed loop system without any overshoot, whereas in terms of the settling time, the DC motor tracks the prescribed reference speed within 0.4 seconds, under simulation conditions, and 0.5 seconds in the experimental case.

As previously mentioned, the proposed implementation, running firstly on the PC, secondly on the real-time target and finally on the FPGA chip, has been tested and validated using the data vectors generated through simulation. In terms of the FPGA implementation, the validation procedure using the simulated data vectors is extremely important, due to the additional changes in the behaviour of the controller, caused by the translation from DBL to FXP representation, that occurred. Additionally, the simulation of FPGA program was also an important action, mainly due to the fact that the program compilation time lasts for approximately 10 minutes. To verify the proposed control solution, three different DC motors from the same power class have been tested. The results showed that the proposed control algorithm running on the FPGA target is robust to changes in motor parameters.

An important parameter characterizing the implementation is the jitter. As it can be seen in Fig. 7, this deviation from true periodicity varies depending on the target. Thus, in the case of the PC, the value is greatly influenced by the tasks that run in parallel with the control application at specific points in time. In this case, the execution time shows variations in the order of milliseconds. The execution time, or the sampling time, for the control loop running on the FPGA and on the real-time controller is constant, with a value of 15msec. However, this later target generates a variation of ±200 μsec.

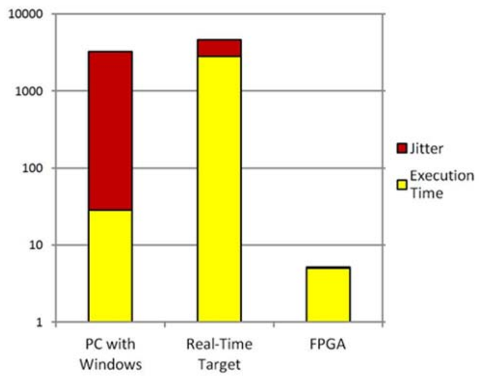

Fig. 7. Execution time and jitter for all targets (usec)

As a conclusion, it is natural that the performance is higher when the execution time is shorter, but, in the same time, jitter is also important, a value as low as possible being desired. Although short sampling times are provided, a jitter value with the same magnitude as the execution time, as can be seen in the case of the PC implementation, negatively affects the entire system, especially for "time critical processes".

VII. CONCLUSIONS

The feasibility of using graphical programming in the design and implementation of control algorithms has been demonstrated for the case of the EPSAC control strategy. Features related to speed, hardware resources, real-time performance and programming aspects have been analyzed and compared using different implementations. The following aspects have also been demonstrated: the portability of the graphical programs on as many industrial standard devices, program scalability providing the possibility of running on resource limited and relatively cheap devices or on high performance systems. The results obtained show that the FPGA implementation stands as a good compromise in terms of computational speed, hardware resource usage, power consumption and real-time performance. Due to these advantages, complex predictive control algorithms may be used in controlling fast dynamic processes. Moreover, the results obtained justify the use of graphical programming techniques, which provide fast synthesis of control algorithms, as well as a shortening of the time to market of dedicated solutions.

Acknowledgement

This work was partially supported by a grant of the Romanian National Authority for Scientific Research, CNDI–UEFISCDI, project number 155/2012 PN-II-PT-PCCA-2011-3.2-0591.

References

[1] J. G. VanAntwerp and R. D. Braatz, "Fast model predictive control of sheet and film processes," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 3, pp. 408 – 417, 2000.

[2] Yang Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267 – 278, 2010.

[3] T. A. Johansen, W. Jackson, R. Schreiber, and P. Tondel, "Hardware synthesis of explicit model predictive controllers," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 1, pp. 191 – 197, 2007.

[4] A. Wills, G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *Control Systems Technology, IEEE Transactions on*, vol. 20, pp. 59 – 71, Jan 2012.

[5] L. Gomes, E. Monmasson, M. Cirstea, and J. J. Rodriguez-Andina, "Industrial electronic control: FPGAs and embedded systems solutions," in *IECON 2013 Conference Proceedings*, pp. 60 – 65, 2013.

[6] C.I. Muresan, E.H. Dulf, R. Both, A. Palfi, M. Caprioru, "Microcontroller Implementation of a Multivariable Fractional Order PI Controller", in *The 9th International Conference on Control Systems and Computer Science*, pp. 44-51, 2013

[7] C.I. Muresan, E.H. Dulf, R. Both, R., "Comparative analysis of different control strategies for a train of cryogenic 13C separation columns", *Chemical Engineering and Technology*, DOI: 10.1002/ceat.201400550

[8] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems – a review," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1824 – 1842, 2007.

[9] E. Monmasson, L. Idkhajine, and M. Naouar, "FPGA-based controllers," *IEEE Industrial Electronics Magazine*, vol. 5, pp. 14 – 26, March 2011.

[10] E. Monmasson and M. Cirstea, "Guest editorial special section on industrial control applications of FPGAs," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1250 – 1252, 2013.

[11] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended Kalman filter," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 3908 – 3918, Oct. 2012.

[12] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 224 – 243, 2011.

[13] C. Sepulveda, J. Munoz, J. Espinoza, M. Figueroa, and F. C. Baier, "FPGA v/s DSP performance comparison for a VSC-based STATCOM control application," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2012.

[14] A. Dinu, M. N. Cirstea, and S. E. Cirstea, "Direct neural-network hardware-implementation algorithm," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1845 – 1848, 2010.

[15] J. Rodriguez-Araujo, J. Rodriguez-Andina, J. Farina, F. Vidal, J. Mato, and M. A. Montealegre, "Industrial laser cladding systems: FPGA-based adaptive control," *IEEE Industrial Electronics Magazine*, vol. 6, pp. 35 – 46, Dec. 2012.

[16] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1810 – 1823, 2007.

[17] S. Kouro, P. Corts, R. Vargas, U. Ammann, and J. Rodriguez, "Model predictive control-a simple and powerful method to control power converters," *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 1826 – 1838, June 2009.

[18] J. Rodriguez, R. M. Kennel, J. R. Espinoza, M. Trincado, C. A. Silva, and C. A. Rojas, "High-performance control strategies for electrical drives: An experimental assessment," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 812 – 820, Feb. 2012.

[19] P. Martin Sanchez, O. Machado, E. Bueno, F. J. Rodriguez, and F. J. Meca, "FPGA-based implementation of a predictive current controller for power converters," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2012.

[20] E. Hartley, J. Jerez, A. Suardi, J. Maciejowski, E. Kerrigan, and G. Constantinides, "Predictive control using an FPGA with application to aircraft control," *Control Systems Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1 – 12, 2013.

[21] R. D. Keyser, "Model based predictive control," *UNESCO Encyclopaedia of Life Support Systems (EoLSS)*, vol. 83, 2003. article 6.43.16.1, Eolss Publishers Co Ltd, Oxford, ISBN 0 9542 989 18-26-34.

[22] T. Orlowska-Kowalska and M. Kaminski, "FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system," *IEEE Transactions on Industrial Informatics*, vol. 7, pp. 436 – 445, Aug. 2011.